CLAIMS

1. One or more computer readable media having stored thereon a plurality of instructions that, when executed by one or more processors, causes the one or more processors to perform acts including:

identifying a plurality of modules in a software program, wherein each module includes a plurality of blocks and wherein the plurality of modules includes checker modules;

for each of the plurality of modules,

generating an original checkpoint value, and

incorporating the original checkpoint value into a checker module; and

for each of the checker modules,

generating a new checkpoint value after the original checkpoint value has been incorporated into the checker module, and

determining a new block to add to the checker module to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker module equals the original checkpoint value for the checker module.

2. One or more computer readable media as recited in claim 1, wherein the incorporating comprises incorporating the original checkpoint value into multiple checker modules.

3. One or more computer readable media as recited in claim 1, wherein the generating comprises:

computing, based on the plurality of blocks of a module, a message authentication code (MAC) value to be used as the checkpoint value for the module.

4. One or more computer readable media as recited in claim 3, wherein the computing comprises:

inputting each of the plurality of blocks of the module into an exclusive-or operator that generates an output value by performing an exclusive-or operation on each block and an encrypted version of the previous output of the exclusive-or operator; and

using, as the message authentication code value, the output value from the exclusive-or operator obtained from inputting the last of the plurality of blocks into the exclusive-or operator.

5. One or more computer readable media as recited in claim 1, wherein the determining a new block comprises:

encrypting the new checkpoint value; and

determining, as the content of the new block, a value equal to the exclusive-or of the encrypted new checkpoint value and the original checksum value.

6. One or more computer readable media as recited in claim 1, wherein the new block does not alter the functionality of the module.

7. One or more computer readable media as recited in claim 1, wherein the new block comprises a data block.

8. One or more computer readable media as recited in claim 1, wherein the plurality of instructions, when executed, further causes the one or more processors to perform acts including adding, prior to generating the new checkpoint value, additional instructions to the module as part of one or more additional blocks, the additional instructions causing the addition of the new block to not alter the functionality of the module.

9. One or more computer readable media as recited in claim 1, wherein the software program further includes a plurality of checkpoints corresponding to the incorporated checkpoint values, wherein each checkpoint identifies when the integrity of the corresponding module is to be verified.

10. A method comprising:

identifying a plurality of segments in an object; and

applying cyclic integrity verification to the object based on the plurality of segments.

11. A method as recited in claim 10, wherein the cyclic integrity verification is applied to the plurality of segments by:

for each of the plurality of segments,

generating an original checkpoint value, and incorporating the original checkpoint value into a checker segment; and

for each of the checker segments,

generating a new checkpoint value after the original checkpoint value has been incorporated into the checker segment,

determining an additional block to be added to the checker segment to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker segment equals the original checkpoint value for the checker segment.

12. A method as recited in claim 10, wherein the cyclic integrity verification is applied to verify the shape of the plurality of segments.

13. A method as recited in claim 10, wherein the cyclic integrity verification is applied to verify the behavior of the plurality of segments.

14. One or more computer-readable memories comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 10.

15. A method comprising:

identifying a plurality of segments in an object;

generating a checkpoint value for each of the plurality of segments;

storing the checkpoint value for each of the plurality of segments in another of the plurality of segments; and modifying each of the plurality of segments so that the addition of the checkpoint value to the segment is offset and the checkpoint value for the segment remains the same.

16. A method as recited in claim 15, wherein the storing comprises storing the checkpoint value into multiple other segments of the plurality of segments.

17. A method as recited in claim 15, wherein the modifying comprises:

computing, based on a plurality of blocks of a segment, a message authentication code (MAC) value to be used as the checkpoint value for the segment; and

determining a new block to add to the segment to offset the stored checkpoint value such that subsequent generation of a checkpoint value for the segment equals the previously generated message authentication code value.

18. A method as recited in claim 17, wherein the computing comprises:

inputting each of the plurality of blocks of the segment into an exclusive-or operator that generates an output value by performing an exclusive-or operation on each block and an encrypted version of the previous output of the exclusive-or operator; and

using, as the message authentication code value, the output value from the exclusive-or operator obtained from inputting the last of the plurality of blocks into the exclusive-or operator.

19. A method as recited in claim 17, wherein the determining a new block comprises:

generating a new checkpoint value based on the plurality of blocks and including the stored checkpoint value;

encrypting the new checkpoint value; and

determining, as the content of the new block, a value equal to the exclusive-or of the encrypted new checkpoint value and the original checksum value.

20. A method as recited in claim 15, wherein the modifying does not alter the functionality of the segment.

21. A method as recited in claim 15, wherein the modifying comprises adding a new data block.

22. A method as recited in claim 15, wherein the object comprises a software program.

23. A method as recited in claim 15, further comprising storing a checkpoint corresponding to each checkpoint value, each checkpoint identifying when the integrity of the corresponding segment is to be verified.

24. One or more computer-readable memories comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 15.

25. A method comprising:

generating a verification value for a first segment of an object;

generating an original verification value for a second segment of the object;

adding, to the second segment, the verification value for the first segment; and

adding an offset value to the second segment so that a newly calculated verification value for the second segment equals the original verification value.

26. A method as recited in claim 25, wherein the generating the verification value for the first segment comprises generating the verification value based at least in part on behavior of the first segment during execution of the first segment.

27. A method as recited in claim 26, wherein the behavior of the first segment during execution includes modification of a register by one or more instructions in the first segment during execution.

28. A method as recited in claim 25, further comprising:

adding, to the first segment, the original verification value for the second segment; and

adding another offset value to the first segment so that a newly calculated verification value for the first segment equals the verification value for the first segment.

29. A method as recited in claim 25, wherein the generating the original verification value comprises:

computing, based on a plurality of blocks of the second segment, a message authentication code (MAC) value.

30. A method as recited in claim 29, wherein the computing comprises:

inputting each of the plurality of blocks of the second segment into an exclusive-or operator that generates an output value by performing an exclusive-or operation on each block and an encrypted version of the previous output of the exclusive-or operator; and

using, as the message authentication code value, the output value from the exclusive-or operator obtained from inputting the last of the plurality of blocks into the exclusive-or operator.

31. A method as recited in claim 25, wherein the adding an offset value comprises:

generating a new verification value for the second segment;

encrypting the new verification value; and

determining, as the offset value, a value equal to the exclusive-or of the encrypted new verification value and the original verification value.

32. A method as recited in claim 25, wherein the offset value does not alter the functionality of the module.

33. A method as recited in claim 25, wherein the offset value comprises a data block.

34. A method as recited in claim 25, wherein the object comprises a software program.

35. A method as recited in claim 25, further comprising storing a checkpoint, corresponding to the verification value, that identifies when the integrity of the first segment is to be verified.

36. A method as recited in claim 35, further comprising storing the checkpoint in the second segment.

37. One or more computer-readable memories comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claim 25.

38. One or more computer-readable media having stored thereon a computer program including:

a plurality of segments, each including one or more checkpoint values to be used to verify the integrity of one or more other segments; and

wherein the plurality of segments further include a plurality of checkpoints that identify a circular ordering of verifying the integrity of the segments.

39. One or more computer-readable media as recited in claim 38, wherein each of the checkpoint values is message authentication code (MAC) value based on the one or more other segments.

40. One or more computer-readable media as recited in claim 38, wherein each of the plurality of segments includes a checkpoint value to be used to verify the integrity of each of the other of the plurality of segments.

41. A production system, comprising:

a memory to store an original program; and

a production server equipped with a cyclic integrity verification protection tool that is used to augment the original program for protection purposes, the production server being configured to parse the original program into a plurality of segments and apply cyclic integrity verification to the plurality of segments.

42. A production system as recited in claim 41, wherein the cyclic integrity verification is applied to the plurality of segments by:

for each of the plurality of segments,

generating an original checkpoint value, and

incorporating the original checkpoint value into a checker segment; and

for each of the checker segments,

generating a new checkpoint value after the original checkpoint value has been incorporated into the checker segment, and determining an additional block to be added to the checker segment to offset the incorporated original checkpoint value such that subsequent generation of a checkpoint value for the checker segment equals the original checkpoint value for the checker segment.

43. A production system as recited in claim 42, wherein the cyclic integrity verification is applied to the plurality of segments by further including a plurality of checkpoints corresponding to the incorporated checkpoint values, wherein each checkpoint identifies when the integrity of the corresponding segment is to be verified.

44. A client-server system, comprising:

a production server to apply cyclic integrity verification to a program to produce a protected; and

a client to store and execute the protected program, the client being configured to evaluate the protected program to determine whether the protected program has been tampered with.